Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе 2**

**Дисциплина**: Вычислительная математика

Вариант 1

Выполнил студент гр. 3530901/90003 В.С. Андрианов

(подпись)

Преподаватель В.Н. Цыган

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Содержание**

1. **Техническое задание**
2. **Цель работы**
3. **Ход выполнения работы**
   1. **Используемая среда разработки и язык программирования**
   2. **Объяснение подхода к решению поставленной задачи**
   3. **Описание входных и выходных данных для работы исследуемых процедур**
   4. **Листинг программы**
   5. **Результат выполнения программы**
4. **Вывод**

**1. Техническое задание**

Составить процедуру формирования матрицы A(N,N), зависящей от параметра X:

Используя эту процедуру, подпрограммы DECOMP и SOLVE, найти для N=5 и значений X = 1.1, 1.001, 1.00001, для каждого X вычислить матрицу R = – E и ее норму.

**2. Цель работы**

Использовать в рабочих целях реализацию базовых подпрограмм, таких как Decomp и Solve.

**3. Ход выполнения работы**

* 1. **Используемая среда разработки и язык программирования**

Для решения поставленной задачи в качестве языка программирования был выбран язык С++. Разработка велась в Clion (среда разработки).

**3.2 Объяснение подхода к решению поставленной задачи**

В начале выполнения работы необходимо составить процедуру, которая формирует матрицу A. Этой задачей занимается процедура makeAMatrix(x), которая на вход принимает только значение x. Далее необходимо найти обратную матрицу . За выполнение этой задачи отвечает процедура makeInverseA(A). На вход процедура makeInverseA(A) получает матрицу, к которой необходимо найти обратную. Процедура makeInverseA(A) использует базовые подпрограммы Decomp и Solve, а также в этой процедуре записывается число обусловленности (после работы подпрограммы Decomp). После нахождения обратной матрицы необходимо создать единичную матрицу, за это отвечает процедура makeEMatrix(). После получения единичной матрицу можно найти матрицу R. Для нахождения матрицы R была реализована процедура makeRMatrix(A, A^-1, E), которая на вход получает матрицу A, обратную матрицу матрице A и единичную матрицу E, и, с помощью вспомогательных процедур, вычисляет матрицу R. Также были реализованы вспомогательные процедуры: makeEmprtyMatrix(), multiply(A, invA), minus(A, B), norm(A), print(matrix). Процедура makeEmprtyMatrix() создает пустую матрицу (размерностью N). Процедура multiply(A, invA) получает на вход две матрицы и перемножает их. Процедура minus(A, B) получает на вход две матрицы и вычитает одну из другой. Процедура norm(A) получает на вход матрицу и рассчитывает ее норму, норма была взята: . Процедура print(matrix) получает на вход матрицу и выводит ее в консоль.

**3.3 Описание входных и выходных данных для работы исследуемых процедур**

**DECOMP(n, A, cond, IPVT, Work)**

**n** – порядок матрицы

**A** – матрица, которую нужно разложить

**cond** – оценка обусловленности

**IPVT** – вектор номеров ведущих элементов

**Work** – вспомогательный вектор, рабочее поле

**SOLVE(n, A, B, IPVT)**

**n –** порядок матрицы

**A –** факторизованная матрица (полученная из DECOMP)

**B –** вектор правых частей

**IPVT –** вектор ведущих элементов

**3.4 Листинг программы**

#include <iostream>  
#include "FORSYTHE.h"  
#include "MATRIX.H"  
  
// Оценка числа обусловленности  
N\_TYPE cond;  
  
// Создание пустой матрицы (ndim,ndim)  
N\_TYPE \*\*makeEmptyMatrix() {  
 auto \*\*Matrix = new N\_TYPE \*[ndim];  
 for (int i = 0; i < ndim ; ++i) Matrix[i] = new N\_TYPE [ndim];  
 return Matrix;  
}  
  
// Умножение двух матриц  
N\_TYPE \*\*multiply(N\_TYPE \*\*A, N\_TYPE \*\*invA) {  
 auto \*\*RES = makeEmptyMatrix();  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) RES[i][j] = 0.0;  
 }  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) {  
 for (int k = 0; k < ndim; ++k) RES[i][j] += A[i][k] \* invA[k][j];  
 }  
 }  
 return RES;  
}  
  
// Вычитание одной матрицы из другой  
N\_TYPE \*\*minus(N\_TYPE \*\*A, N\_TYPE \*\*B) {  
 auto \*\*RES = makeEmptyMatrix();  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) RES[i][j] = A[i][j] - B[i][j];  
 }  
 return RES;  
}  
  
// Создание матрицы A  
N\_TYPE \*\*makeAMatrix(double x) {  
 auto \*\*A = makeEmptyMatrix();  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) {  
 if (i == j) A[i][j] = 1 + x;  
 else A[i][j] = j + 1.0;  
 }  
 }  
 A[0][0] = 1.0;  
 return A;  
}  
  
// Создание единичной матрицы  
N\_TYPE \*\*makeEMatrix() {  
 auto \*\*E = makeEmptyMatrix();  
 for (int i = 0; i < ndim ; ++i) {  
 for (int j = 0; j < ndim; ++j) {  
 if (i == j) E[i][j] = 1.0;  
 else E[i][j] = 0.0;  
 }  
 }  
 return E;  
}  
  
// Создание обратной матрицы A  
N\_TYPE \*\*makeInverseA(N\_TYPE \*\*A) {  
 N\_TYPE Work[ndim], subMatrix[ndim][ndim];  
 int IPVT[ndim];  
 auto \*\*invA = makeEmptyMatrix();  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) subMatrix[i][j] = A[i][j];  
 }  
  
 decomp(ndim, subMatrix, &cond, IPVT, Work);  
  
 for (int i = 0; i < ndim; ++i) {  
 N\_TYPE B[ndim]{};  
 B[i] = 1.0;  
 solve(ndim, subMatrix, B, IPVT);  
 for (int j = 0; j < ndim; ++j) invA[j][i] = B[j];  
 }  
  
 return invA;  
}  
  
// Создание матрицы R  
N\_TYPE \*\*makeRMatrix(N\_TYPE \*\*A, N\_TYPE \*\*invA, N\_TYPE \*\*E) {  
 N\_TYPE \*\*R;  
 N\_TYPE \*\*AinvA = multiply(A, invA);  
 R = minus(AinvA, E);  
 return R;  
}  
  
// Вычисление нормы матрицы  
double norm(N\_TYPE \*\*A) {  
 double norm = 0;  
 for (int i = 0; i < ndim; ++i) {  
 for (int j = 0; j < ndim; ++j) norm += sqrt(pow(A[i][j], 2));  
 }  
 return norm;  
}  
  
// Вывод матрицы в консоль  
void print(N\_TYPE \*\*matrix) {  
 for (int i = 0; i < ndim ; ++i) {  
 for (int j = 0; j < ndim; ++j) printf("%.10lf ", matrix[i][j]);  
 printf("\n");  
 }  
}  
  
int main() {  
 double x = 1.01;  
 N\_TYPE \*\*A = makeAMatrix(x);  
 N\_TYPE \*\*invA = makeInverseA(A);  
 N\_TYPE \*\*E = makeEMatrix();  
 N\_TYPE \*\*R = makeRMatrix(A, invA, E);  
 printf("A: \n");  
 print(A);  
 printf("\nA^-1: \n");  
 print(invA);  
 printf("\nR: \n");  
 print(R);  
 double norm1 = norm(A);  
 printf("\n x = %lf\n", x);  
 printf("cond = %lf\n", cond);  
 printf("norm = %lf", norm1);  
}

**3.5 Результат выполнение программы**

**A –** составленная матрица A

**A^-1 –** обратная матрица матрицы A

**R –** матрица, полученная после выполнения действий – E

**x** – точка, по которой строится матрица A

**cond** – число обусловленности

**norm** – норма матрицы R ()

В результате выполнения программы должна получиться матрица R. Элементы матрицы R должны быть равны нулю или близки к этому значению, тогда полученная обратная матрица была вычислена верно (при умножении матрицы на ее обратную получается единичная, и из единичной вычитается матрица E, поэтому R должна получиться нулевой). Так как элементы матрицы R равны нулю (или близки к этому значению), то и норма матрицы R должна быть равна нулю (или близка к этому значению).

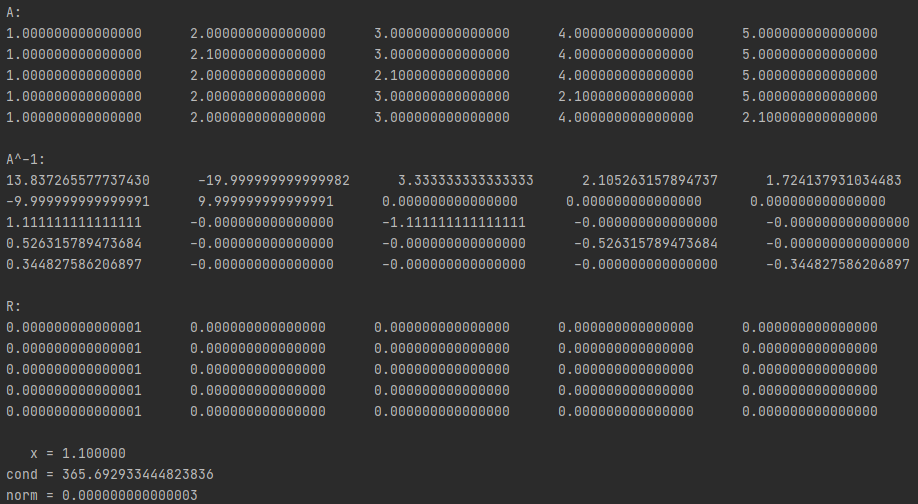


Рис. 1 x = 1.1

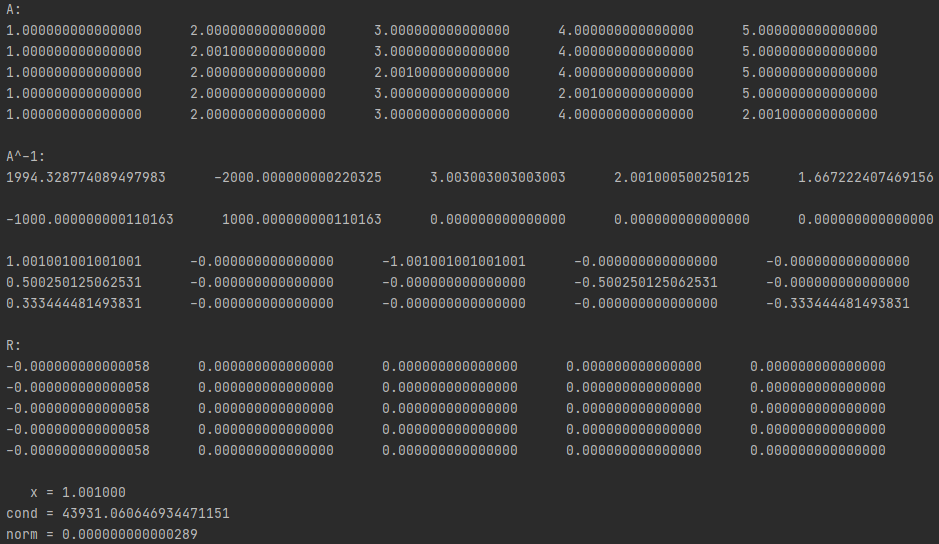


Рис. 2 x = 1.001

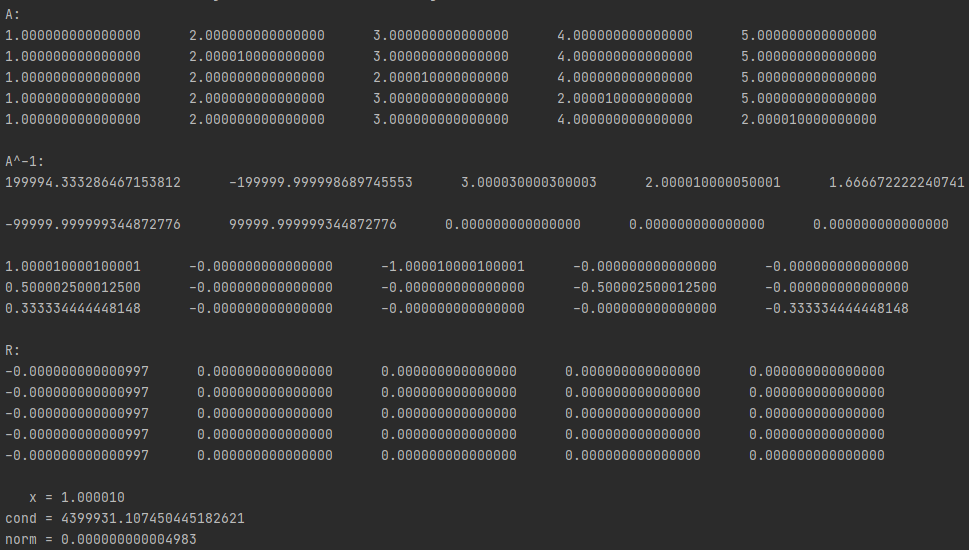


Рис. 3 x = 1.00001

Зависимость нормы матрицы от числа обусловленности матрицы A была рассмотрена с помощью графика. График был построен по результатам работы от x = 1.1, x = 1.01, x = 1.001, x = 1.0001, x = 1.00001.

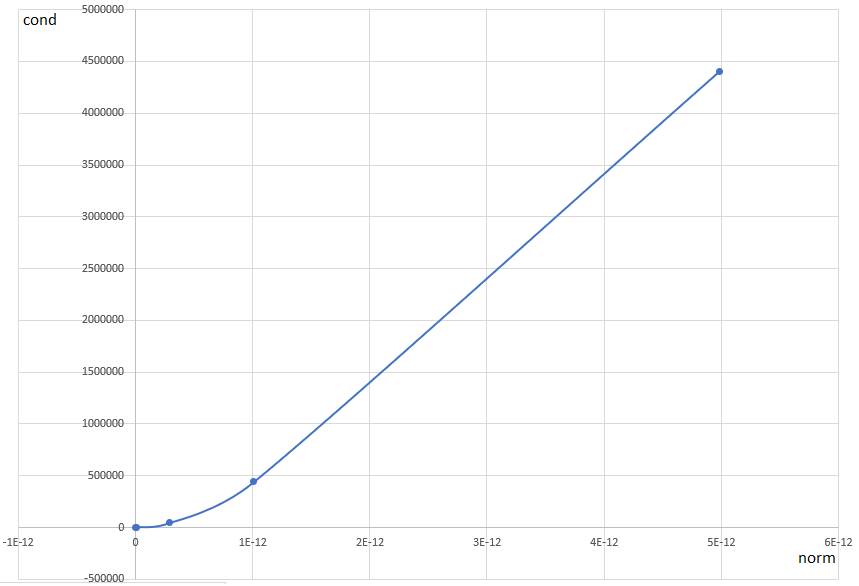


Рис. 4 График зависимости нормы от числа обусловленности матрицы A

Из рисунка 4 видно, что норма почти линейно зависит от числа обусловленности.

**4. Вывод**

В ходе выполнения работы было проведено исследование работы процедур Decomp и Solve.

В результате использования этих процедур были получены обратные матрицы от матриц A. Полученные матрицы были вычислены довольно точно, об этом можно судить по результатам полученных матриц R и их нормам.

Стоит отметить, что чем больше число обусловленности матрицы (cond), тем ближе матрица к вырожденной, то есть, тем сложнее посчитать к ней обратную матрицу. Из проделанной работы видно, что вырожденность матрицы зависит от x, об этом можно судить по результатам работы написанной процедуры.

Также, с помощью графика, была исследована зависимость нормы матрицы от числа обусловленности матрицы A. В результате можно сделать вывод, что зависимость почти линейная.